Lab-8

Submitted by:- Manav Dhiman

Roll no:- 19103063

Q1 Write a program to create two threads for printing all palindrome numbers between 1 to 100 using thread class and implementing a runnable interface.

class pali implements Runnable {

    Thread t;

    pali(String name){

        t=new Thread(this,name);

    }

    public void run(){

        System.out.println("thread is created ");

    }

    boolean check(int n){

        int rev=0,r;

        int cur=n;

        while(n>0){

            r=(n%10);

            rev=rev\*10+r;

            n/=10;

        }

        if(rev==cur){

            System.out.print(t.getName()+" Thread -> ");

            return true;

        }

        return false;

    }

};

class Ques1{

    public static void main(String agrs[]){

        pali p1 = new pali("First");

        pali p2 = new pali("Second");

        for(int i=1;i<=100;){

            if(p1.check(i)) System.out.println(i);

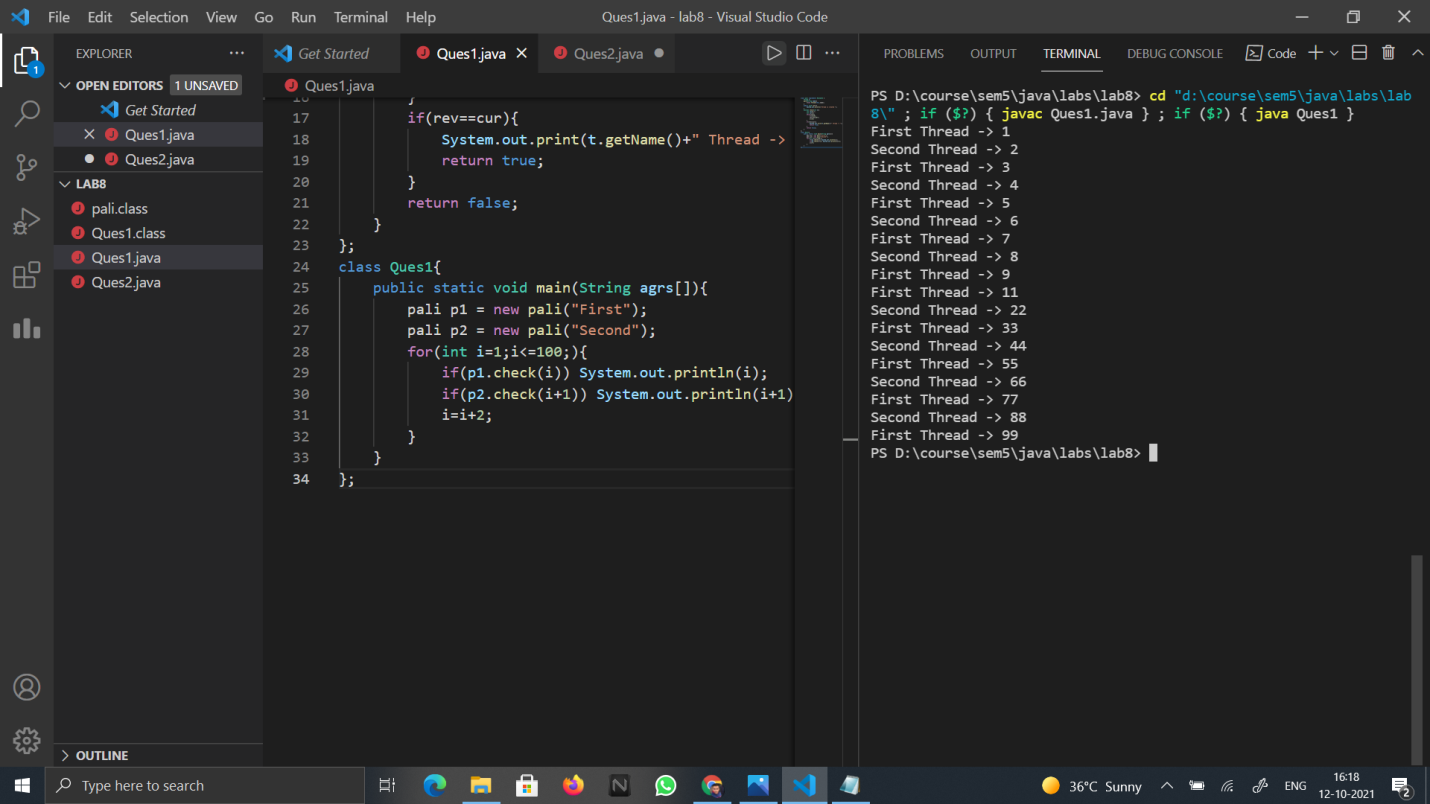
            if(p2.check(i+1)) System.out.println(i+1);

            i=i+2;

        }

    }

};



Q2 Write a Java program to create three threads by extending Thread class in such a way that the first thread is finding the multiplication of two matrices, second is finding the prime numbers between 1 to 100000 and print the counting, and third is to find GCD of some integers values passed through a large integer array.

class Multimat extends Thread{

    Multimat(){

        System.out.println("Matrix multiplication thread: ");

        start();

    }

    public void run(){

        int n=4,m=3,x=3,y=4;

        int a[][] = { { 1, 1, 1 },

                      { 2, 2, 2 },

                      { 3, 3, 3 },

                      { 4, 4, 4 } };

        int b[][] = { { 1, 1, 1, 1 },

                      { 2, 2, 2, 2 },

                      { 3, 3, 3, 3 } };

        int c[][] = new int[n][y];

        for(int i=0;i<n;i++){

            for(int j=0;j<y;j++){

                for(int k=0;k<m;k++){

                    c[i][j]+=a[i][k]\*b[k][j];

                }

            }

        }

        for(int i=0;i<n;i++){

            for(int j=0;j<y;j++){

                System.out.print(c[i][j]+" ");

            }

            System.out.println();

        }

    }

}

class prime extends Thread{

    prime(){

        System.out.println("Prime number thread");

        start();

    }

    boolean thisprime(int n){

        for(int i=2;i\*i<=n;i++){

            if(n%i==0) return false;

        }

        return true;

    }

    public void run(){

        int count=0;

        for(int i=1;i<=100000;i++){

            if(thisprime(i)){

                System.out.println("Prime ->"+i);

                count++;

            }

        }

        System.out.println("Total Primes= "+count);

    }

}

class Gcd extends Thread{

    Gcd(){

        System.out.println("GCD thread");

        start();

    }

    int gcd(int a,int b){

        if(b==0) return a;

        return gcd(b,a%b);

    }

    public void run(){

        int a=12556,b=6458;

        System.out.println("Gcd of "+a+" and "+b+" is ="+gcd(a,b));

    }

}

class Ques2{

    public static void main(String agrs[]){

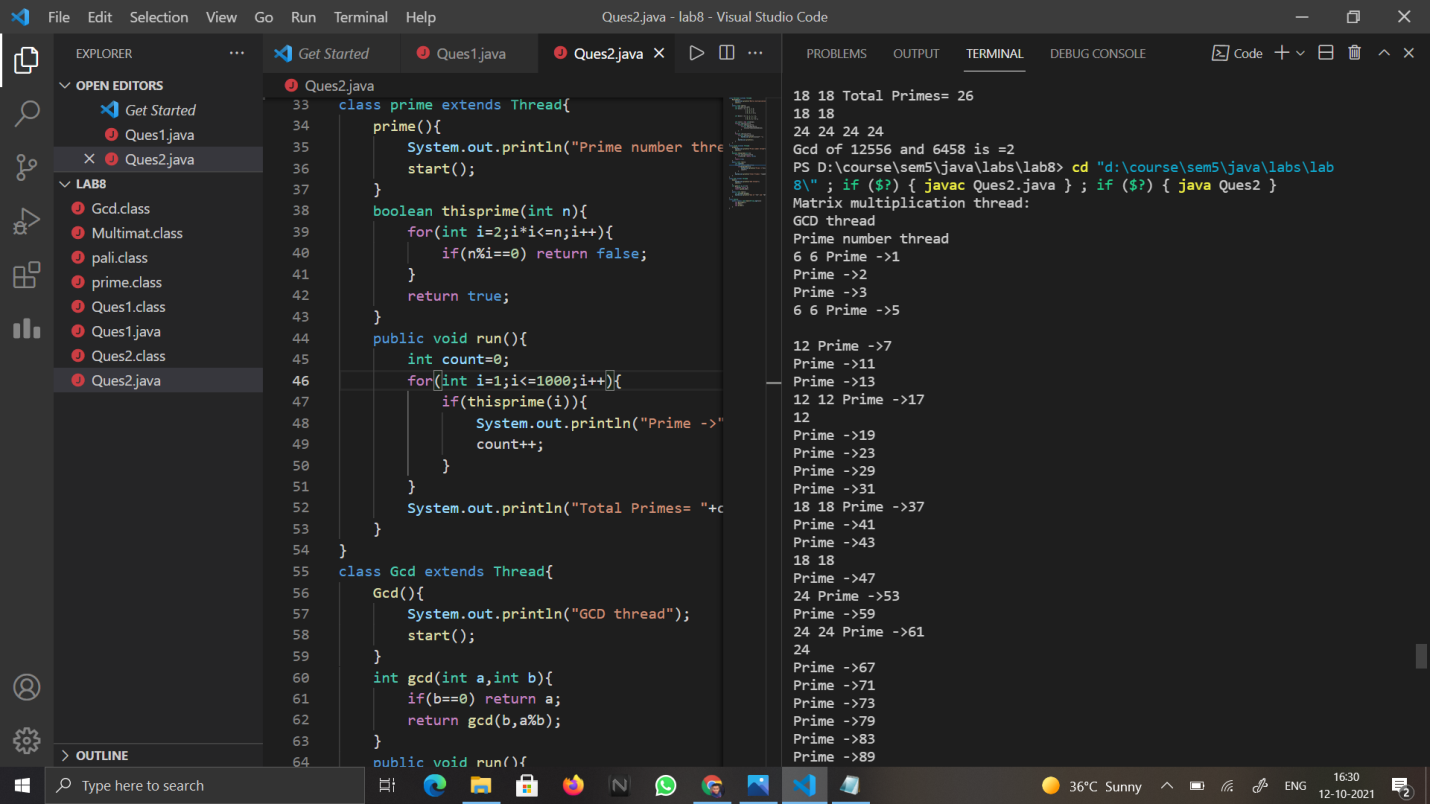
        new Multimat();

        new Gcd();

        new prime();

    }

}



Q3. Write a java program to create three threads by implementing a runnable interface, one is main thread and two other are (child) threads called by main threads in such a way that main thread always terminates after the termination of child threads.

class childt implements Runnable{

    Thread t;

    childt(String s){

        t=new Thread(this,s);

        t.start();

    }

    public void run(){

        try{

            for(int i=0;i<10;i++){

                System.out.println(t.getName() +" "+ i);

                t.sleep(520);

            }

        }

        catch(InterruptedException e){

            System.out.println(e);

        }

        System.out.println(t.getName() + " run is over now");

    }

};

class Ques3{

    public static void main(String args[]){

        childt c1 = new childt("child 1");

        childt c2 = new childt("child 2");

        try{

            while(c1.t.isAlive() || c2.t.isAlive()){

                System.out.println("Child threads are alive");

                Thread.sleep(1000);

            }

        }

        catch(InterruptedException e){

            System.out.println("Main thread interupted by "+e);

        }

        System.out.println("Main thread going to die");

    }

}

